Aalto University

Authentication, naming and pairing

Tuomas Aura

CS-E4300 Network security

Aalto University, autumn 2017

Authentication issues beyond protocols

- What is hard about authentication in a network?
  - Authentication protocol design
  - Knowing who you want to talk to
  - Establishing initial knowledge and trust
- For authenticated key exchange, we usually need
  1. Names or other identifiers
  2. "Root of trust" i.e. prior knowledge of and trust in something
- Examples:
  - TLS: DNS name + certification authority (CA) public key
  - Kerberos: username and AC address + passwords
  - Cellular networks: IMSI + shared key on SIM

IDENTIFIERS = NAMES

What is an identifier?

- In this course, we are interested in identifiers for active entities such as users, computers, communication endpoints, and network elements
- Network protocol designers can define new identifier spaces as needed
  - Typical identifier space is just a set of strings, e.g. MAC address or NAI
  - Sometimes identifier space is a set of more complex expressions, such as domain names, X.500 names (ASN.1)
  - Things that need to be defined: identifier allocation, mapping between the identifiers and communicating entities
- Some identifiers are called "names" if they look and feel like a name to human users
  - This lecture uses the words identifier and name interchangeably
  - Identifier space = name space

Endpoint names

- Authentication and integrity depend on names
- Each protocol layer has its own way of naming endpoints:
  - Ethernet (MAC) addresses in the link layer (e.g. 00-B0-D0-05-04-7E)
  - NAI for network users
  - IP address in the network layer (e.g. 157.58.56.101)
  - TCP port number + IP address
  - DNS or NetBIOS name in the higher layers (e.g. smtp.aalto.fi)
  - URI in web pages and services (e.g. http://www.example.org/myservice)
  - Email address for email users
  - Usernames in online services

Using identifiers

- What architectural entities are being named?
- How are the names and other identifiers allocated?
  - Authority, auto-configuration, ...
- What is the scope of the identifiers?
- Is there a one-to-one mapping between entities and identifiers?
  - Are the identifiers unique within their scope, or can accidental or intentional conflicts (collisions) arise?
  - Could identifiers be shared?
  - Could the same entity have multiple names? Is one of them canonical?
  - Is the mapping static or dynamic?
- How does one reach the owner of a name?
  - Resolving name to the name space of the layer below
  - Routing and data delivery
- How to convince others that this is your name?
  - Authentication, authorization, name ownership
- Secure naming is a difficult and often leads to vulnerabilities

ROOTS OF TRUST

Typical roots of trust for authentication

- Prior knowledge of cryptographic keys:
  - Known public keys
  - Shared master key, e.g. 128-bit shared key
  - Shared weak shared secret, e.g. password (much harder to build secure protocols)
- Trusted third parties (TTP):
  - Certification authority (CA)
  - Online trusted third party, e.g. RADIUS or Kerberos server
  - The words authority and trusted party are often used interchangeably, but it is important to understand the difference!
- Trusted hardware and applications
  - Secure cryptoprocessor, e.g. smart card
  - Trusted execution environment and attestation of its state

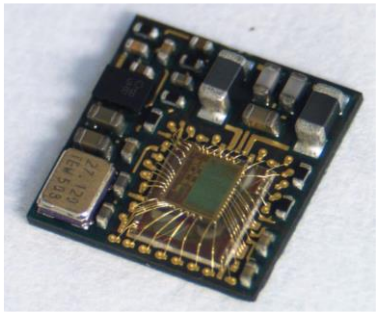
What else can be trusted?

- Secure channels
  - Secure out-of-band channel – authentic and/or secret channel that is not vulnerable to sniffing or spoofing
  - Multiple independent channels – to reduce probability of security failure
  - Location-limited channels – if attacker is unlikely to be in the right place at the right time
  - Human voice and video – hard to spoof with today's technology
- Attempts to avoiding trust and prior knowledge completely
  - Opportunistic key exchange
  - Self-certifying identifiers
  - Context-based security (a kind for secure channel)
  - Proof of work
  - Consensus in a P2P network (a kind of TTP)

SECURE HARDWARE

Secure cryptoprocessor

- Secure hardware can store cryptographic keys
  → keys cannot be leaked by software
- Examples:
  - SIM card
  - Finnish identity card (eID)
  - DESFire smart card, DESFire SAM
  - IBM 4758 cryptographic coprocesso
  - Trusted platform module (TPM)
  - Trusted execution environment



Trusted execution environment

- Isolated computing environment, typically built into the main CPU
  - Protects any computation from software attacks
- Intel TXT, ARM TrustZone, Intel SGX, Global Platform
- Example uses:
  - Storage for cryptographic keys or login credentials
  - Stored value applications e.g. mobile ticketing
  - DRM i.e. copy protection
- Remote attestation: communication endpoint can prove to a remote server that it is running the unmodified application
  - Possible to prove configuration even anonymously

SECURE CHANNELS

Secure out-of-band (OOB) channel

- The OOB channel may be authentic, secret, or both
- Traditional OOB channels:
  - User or system administrator configuring secret keys
  - Armed courier, diplomatic mail etc.
- OOB channels in device pairing:
  - Touching, touching with “magic wand”
  - User-verified key exchange
  - Sound (hard to spoof without detection)
  - User-transferred short secret or authentication code
  - Synchronous user input
  - Secret shared data from context sensing

Two-channel authentication

- Authentication over two independent channels → attacker needs to compromise both
- Applications:
  - Text message to confirm online bank transaction
  - Two-factor authentication by Google, Microsoft, Facebook etc.
- Common problem: channels rarely are completely independent
  - Internet connections promise best effort delivery, not independent routes
  - app on phone and text message to phone both depend on the security of the phone

Location-limited channel

- Some channels are relatively secure if the attacker is not in the room at the time of the key exchange
- Examples of location-limited channels:
  - NFC, short-distance and directional radio, Bluetooth, camera, infrared, visible light, audio
- Caveats:
  - Audio bugs and cameras get ever smaller
  - Computers and phones can be used for spying
  - Information may leak further than you think (e.g. radio signals, displays, keyboards)

Human voice or video authentication

- It is still difficult to spoof humans
  - Remember the Turing test for artificial intelligence
- Examples:
  - Personal meeting
  - Cryptophone – human voice verification of the key exchange (How to do this for Diffie-Hellman key exchange?)
- Caveats:
  - Computers are getting better at processing live voice and video
  - Meeting a person does not guarantee they are trustworthy

AUTHENTICATION WITHOUT TRUST OR PRIOR KNOWLEDGE

Opportunistic security, leap of faith

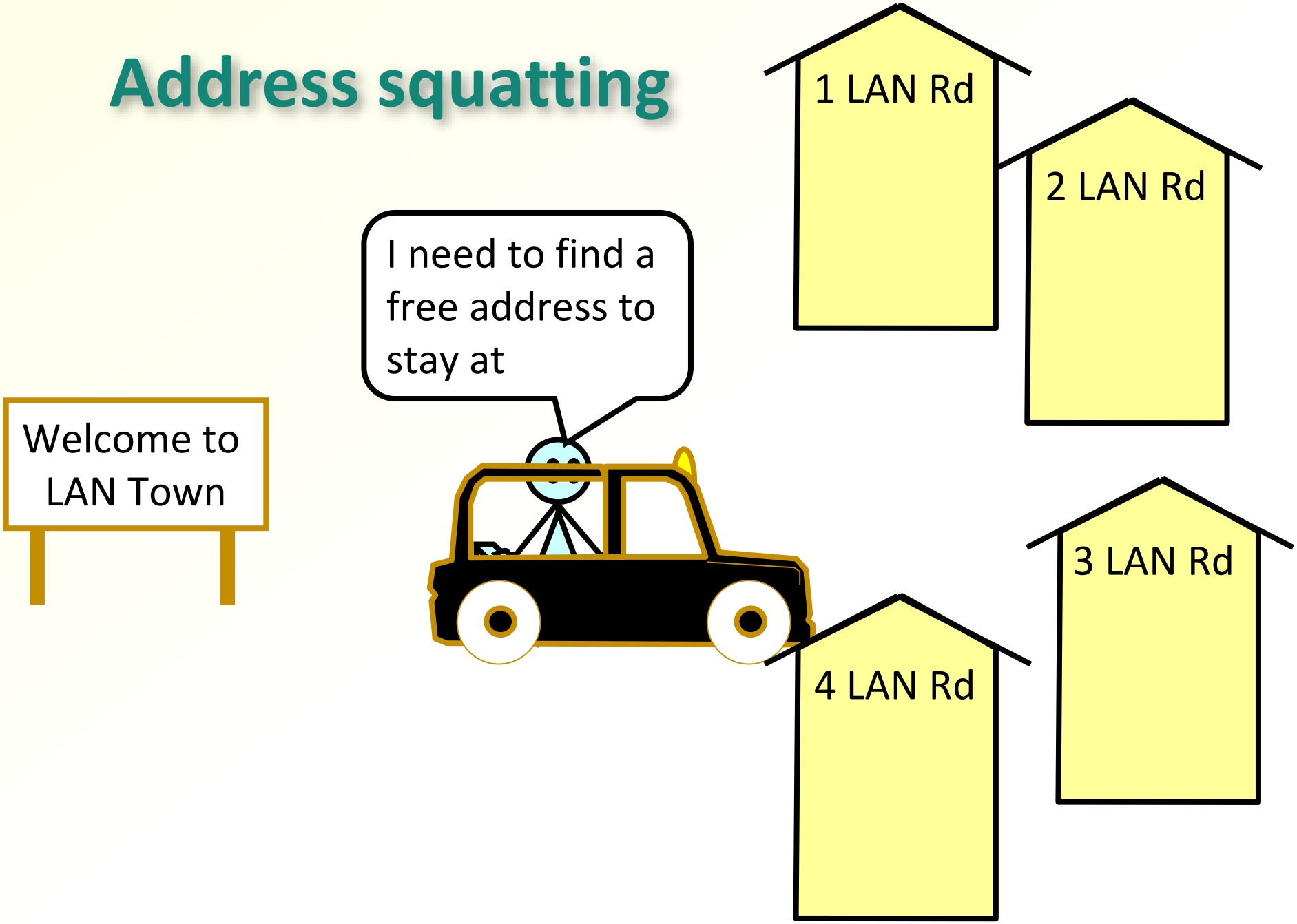
- Opportunistic encryption: encrypt without authentication
  - Opportunistic IPsec (RFC 4322)
  - STARTTLS with self-signed certificates
- In leap of faith, the first key exchange is unauthenticated, then keys remembered
  - Secure Shell (SSH) – first introduced leap of faith
  - HTTP strict transport security (HSTS) with self-signed certificates (self-signed not allowed in RFC 6797)
- Resurrecting ducking model for device pairing
  - Device associates to the first master it sees after reset
- Idea: attacker is unlikely to be always present
- Dangers:
  - Leap of faith cannot be reused to recover from failure after the first authentication (e.g. changed SSH host key)
  - Must be started by a human, not triggered automatically

Self-certifying identifiers

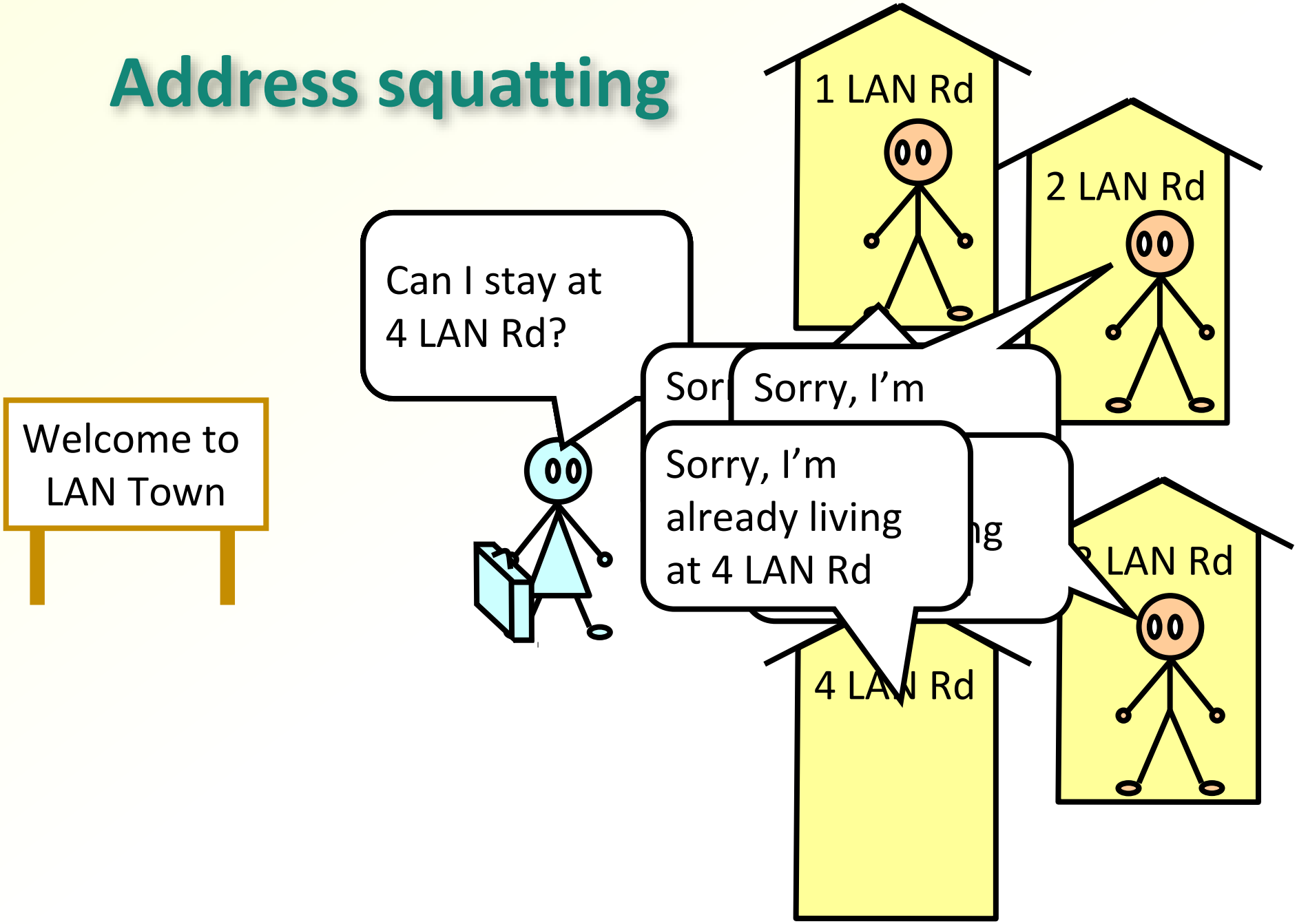
- Public key or its hash as entity identifier
- Examples:
  - Self-signed certificate
  - Cryptographically generated IPv6 addresses (CGA)
  - HIP host identity (RFC 4423)
- Hash of the data as object identifier
- Examples:
  - Self-certifying file system
  - BitTorrent and other P2P systems

CASE 3: IPV6 ADDRESS OWNERSHIP AND SQUATTING

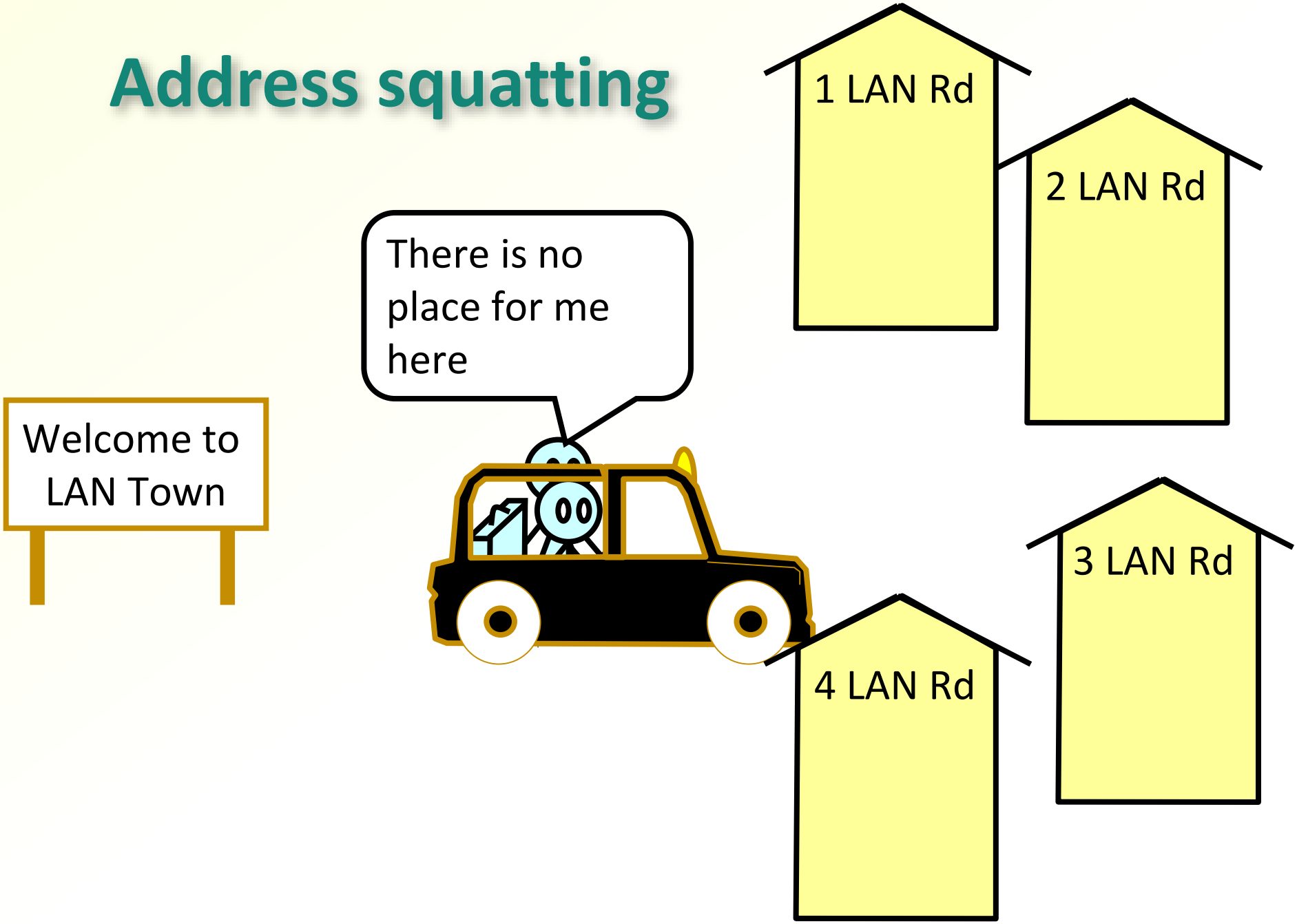
Address squatting



Address squatting



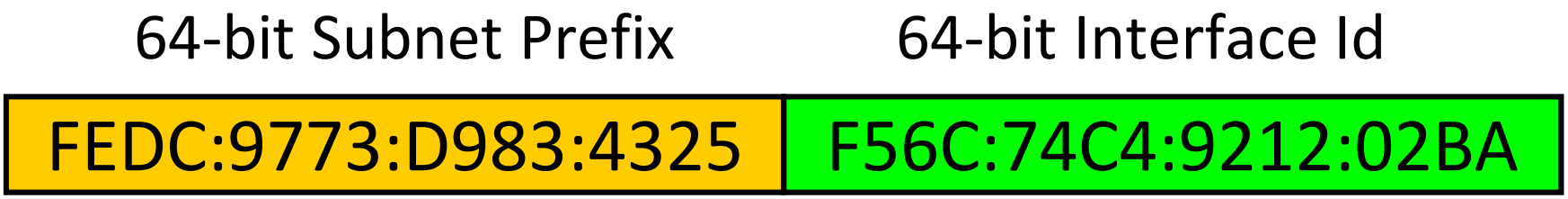
Address squatting



Addresses and identifier allocation

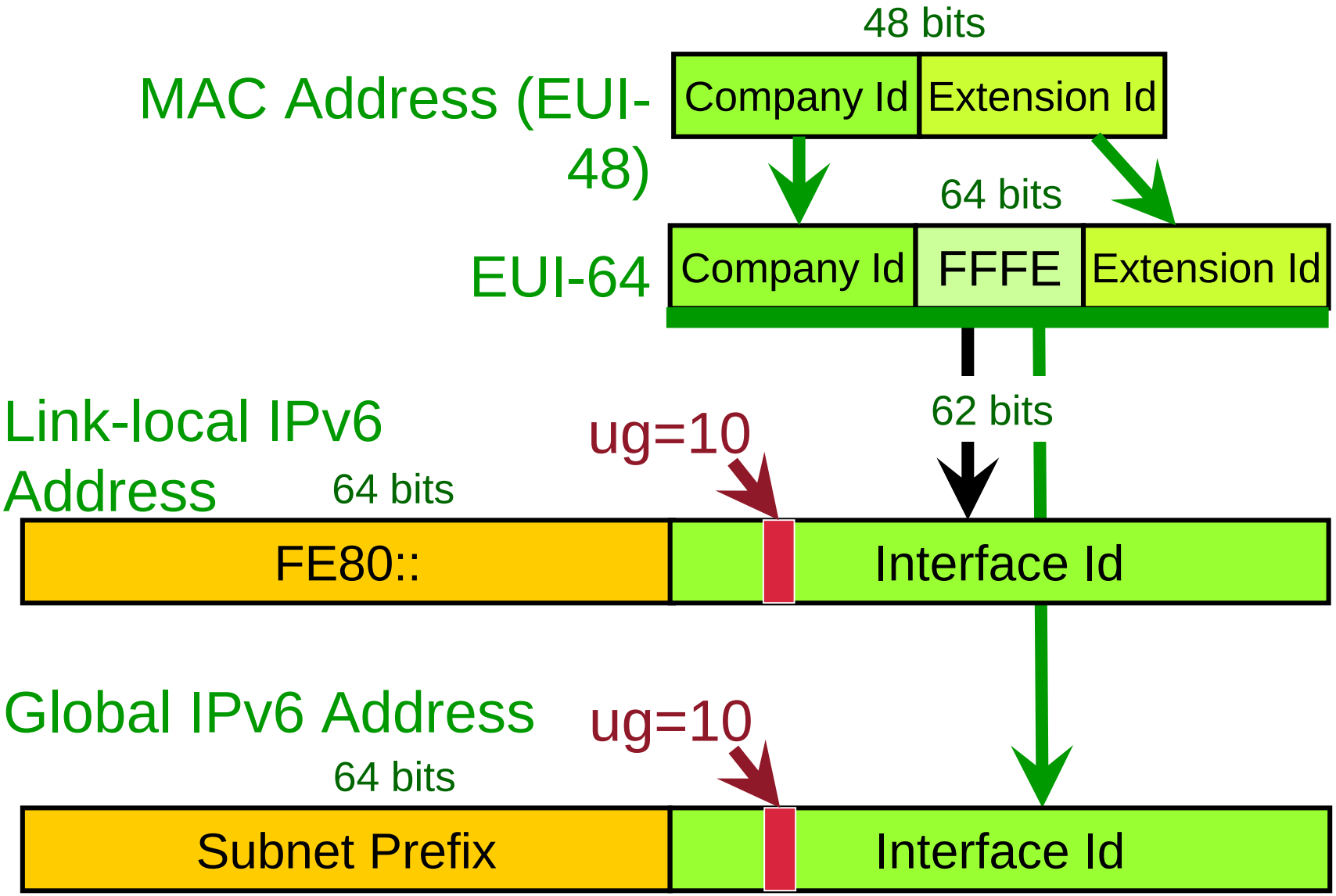
- Methods for allocating IP addresses and other unique identifiers:
  - Static allocation — IP addresses, MAC addresses
  - Stateful configuration by a server — DHCP
  - Autoconfiguration — IPv6 addresses
- Autoconfiguration requires least infrastructure and administration, is most scalable, and is suitable for ad-hoc and mobile-access networks
- Autoconfiguration is also most vulnerable to attacks like address squatting

IPv6 address

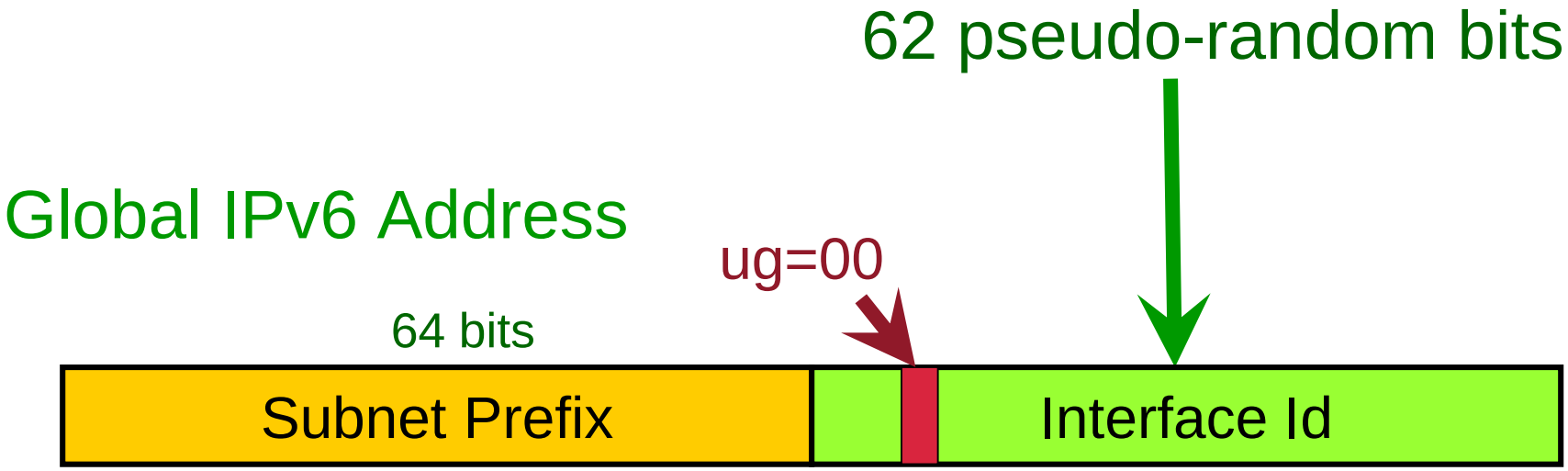


- Nodes attached to the same gateway router have the same subnet prefix but different interface ids
- Subnet prefix is used for routing
- 62 bits of the interface id can be chosen in random (2 bits have a special meaning)

Stateless autoconfiguration



Address privacy extensions (RFC 4941)



- The interface identifier is randomized to enhance user privacy: servers on the internet cannot recognize the client machine by its IPv6 address

Configuring IPv6 addresses

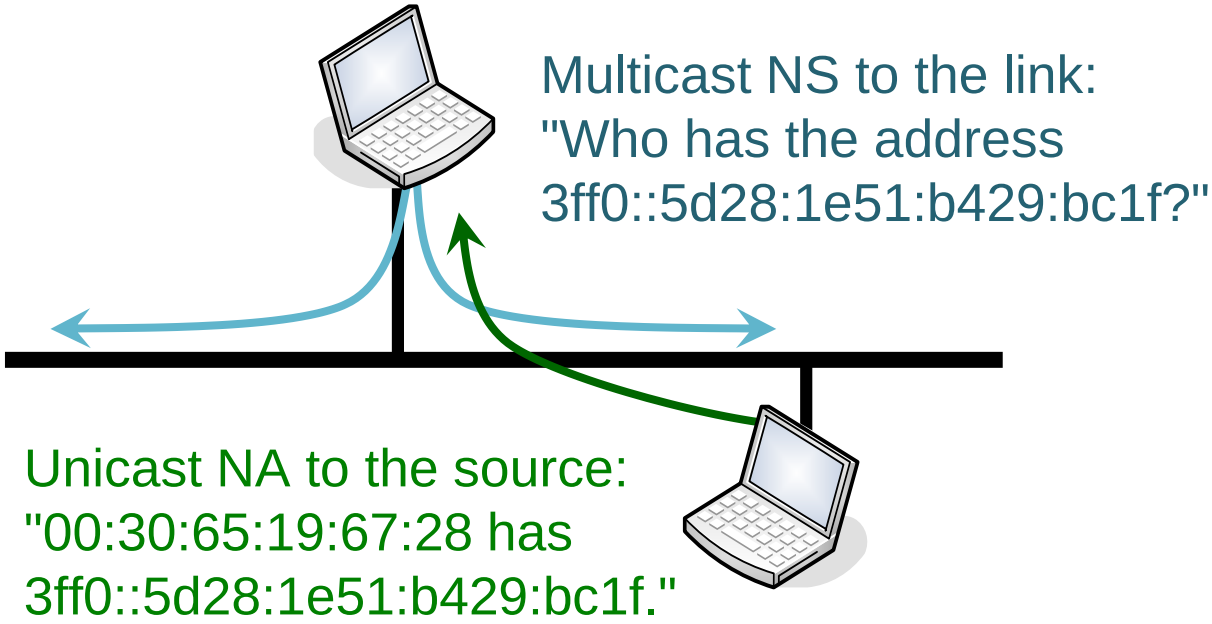
- Host's addresses [RFC 4291]:
  - Zero or more global addresses:
    subnet prefix | interface identifier
  - At least one link-local address for each interface:
    FE80::0 | interface identifier
- Router has one link-local address for each interface
- Stateless address autoconfiguration [RCF 4862]:
  - Host creates a link-local address and performs duplicate address detection (DAD)
  - Host performs router discovery to obtain router addresses and subnet prefixes; it chooses which one(s) to use
  - Host creates a global address for each prefix and performs DAD (some implementations don't)
- Neighbor discovery [RFC 4861] maps IP addresses to MAC addresses

Uniqueness of addresses

- EUI-64 addresses are supposed to be unique because MAC addresses are
  → Address collision is an unrecoverable error. Give up and report failure
- IPv6 address privacy extensions have random interface identifiers, which may sometimes collide
  → Try different random values and perform DAD. After a few collisions, give up and report error
  (How likely is a collision?)
- DHCPv6 can be used to assign addresses instead of stateless autoconfiguration
- In all cases, duplicate address detection is mandatory

IPv6 equivalent of ARP

Neighbor discovery

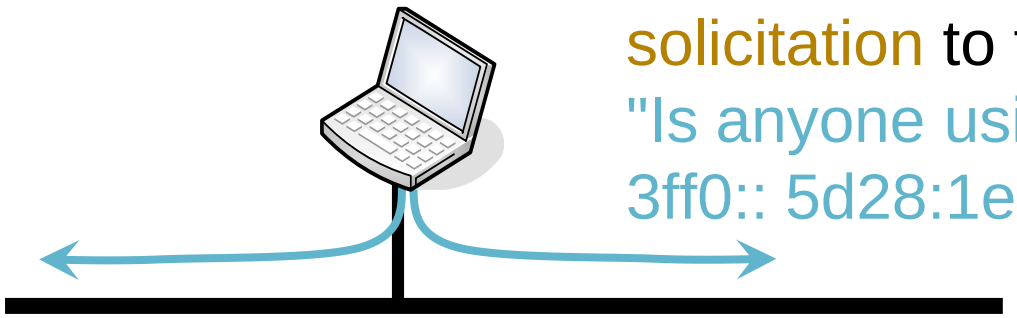


- Multicast neighbor solicitation (NS), unicast neighbor advertisement (NA)
- Also unsolicited multicast NA

Duplicate address detection (DAD)

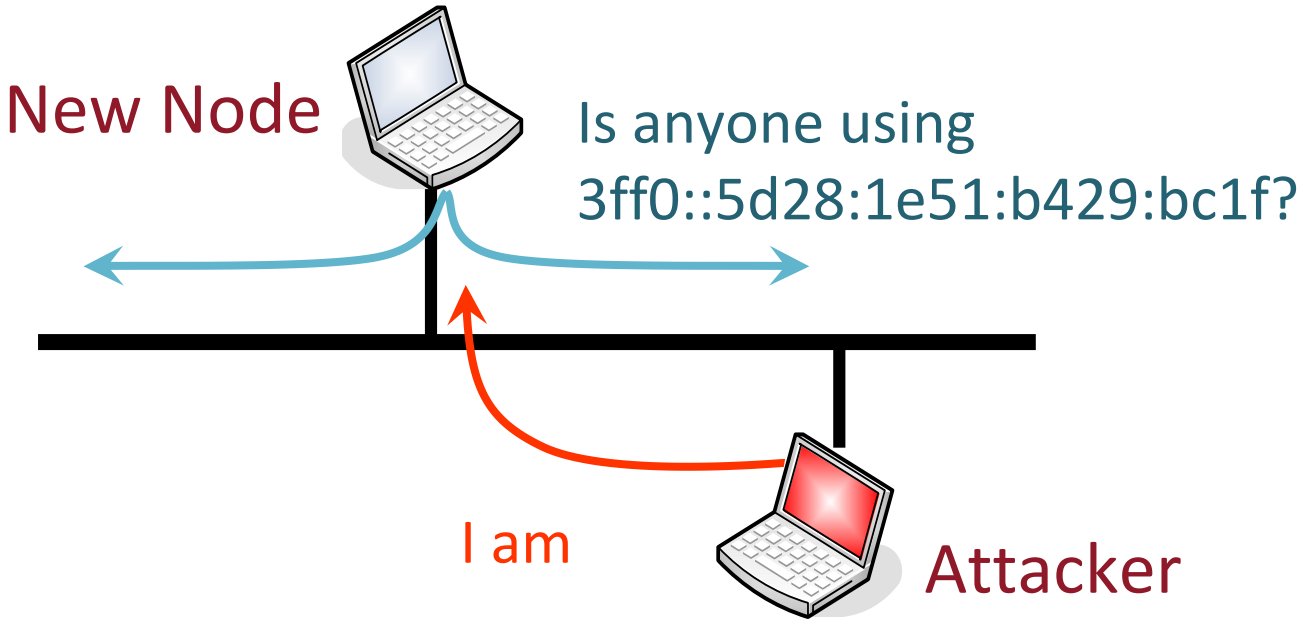
- During address autoconfiguration, DAD is required for each unicast address to detect accidental address collisions and administrative errors

1. Pick an address: 3ff0::5d28:1e51:b429:bc1f
2. Multicast a neighbor solicitation to the link "Is anyone using 3ff0:: 5d28:1e51:b429:bc1f?"



No answer → address ok

DAD address squatting



- Attacker responds to every neighbor solicitation (NS) from the new node with a neighbor advertisement (NA)

  → New node cannot find a free address

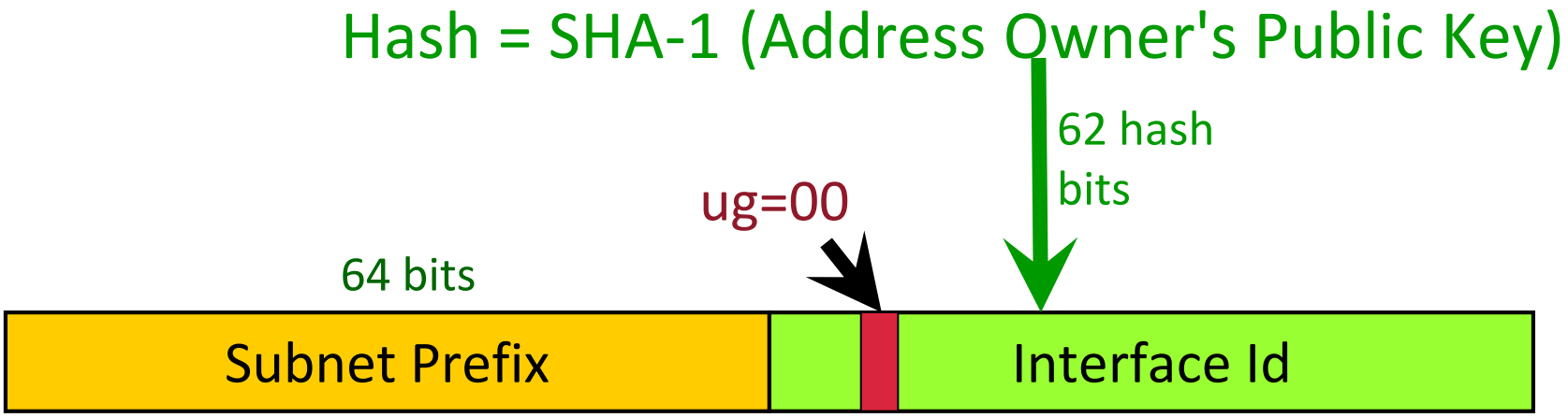
CRYPTOGRAPHICALLY GENERATED ADDRESSES (CGA)

Address ownership

- Needed: a mechanism for proving address ownership
- Potential uses:
  - Preventing DAD address squatting
  - Preventing spoofing of neighbor advertisements in neighbor discovery
  - Authenticating Mobile IPv6 binding updates
  - Authenticating ICMPv6 error messages
  - Exchanging keys for opportunistic IPSec

Cryptographically generated address (CGA)

- The interface identifier contains the address owner's public signature key → can sign messages sent from the address
  - CAM proposal for Mobile IPv6 [O'Shea & Roe 2000]



Proof of address ownership

- Node sends the public key and a signed message from the CGA address
- Receiver
  - Recomputes the hash of the public key
  - Compares the hash with the with the interface id of the source address
  - Verifies the signature using the public key

  → Receiver knows that the message was sent by the owner of the source address
- CGA-signing can prevent spoofing of IP-layer signaling messages such as neighbor advertisements

Countering dictionary attacks

- Attacker could create a database of all (or most) interface identifiers and corresponding public keys
- Solution: include the subnet prefix as “salt” in the hash input
- However, link-local addresses still vulnerable and every IPv6 node needs one

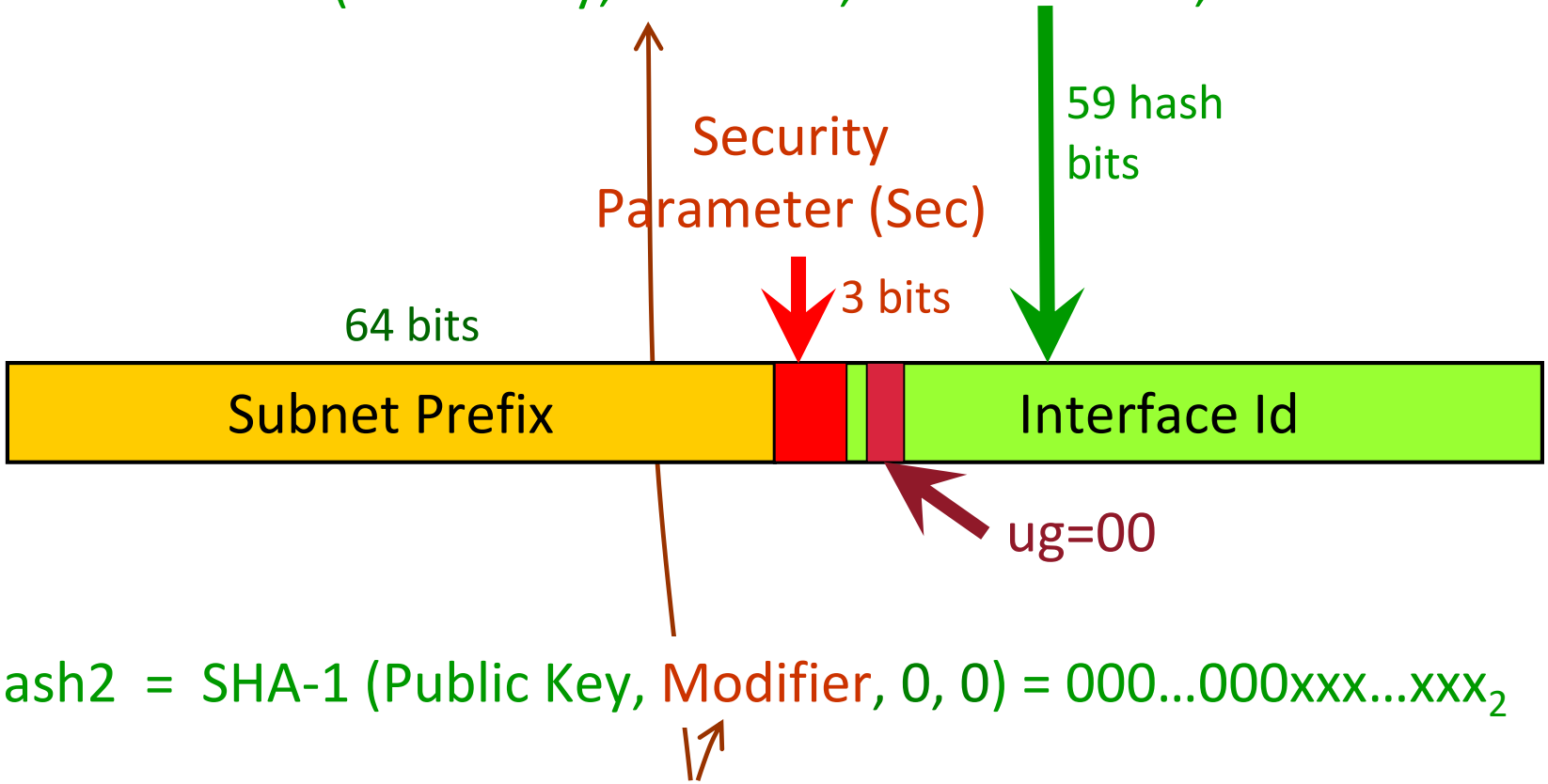
Hash extension

- The hash in CGA is at most 62 hash bits → vulnerable to brute-force attacks in the foreseeable future
- Moore's law (one variation): CPU speed doubles every 18 months → one bit of hash strength lost
  → in about 30 years, CGA might be useless
  - Already too weak for strong authentication but still ok for DoS protection
- Solution:
  - Increase artificially the cost of a brute-force attack
  - Cost of creating a CGA will increase by the same factor
  - Allow CGA creator to decide how much extra strength is needed
  - Cost of using CGA (signing and verifying) will stay constant

Standard CGA address format

[RFC 3972]

Hash1 = SHA-1 (Public Key, Modifier, Subnet Prefix, Collision Count)



Hash2 = SHA-1 (Public Key, Modifier, 0, 0) = 000…000xxx…xxx$_2$

Modifier must be chosen so that
Hash2 begins with 16*Sec zero bits.

Bidding down problem

- Cannot require all Internet nodes to have CGA addresses. Which addresses are CGA and which are not?
- Cannot trust the address owner to tell. Attacker can claim that it is not using CGA even when it is
- Solutions:
  - Our proposal, not accepted in IETF: use an unused combination of “g” and “u” bits (g=1 and u=1) in the interface id as a type tag for CGAs
  - Current solution: Prioritize CGAs. CGA-signed data will overwrite unsigned data (e.g. in the neighbor cache) but not the other way

CGA limitations

- DNS names must be mapped to IP addresses
  → CGA-based authentication prevents spoofing of source IP addresses; it does not prevent DNS spoofing
- Authenticates the interface identifiers only, not the subnet prefix (=location in the network topology)
- CGA-based authentication prevents spoofing of someone else's IP address. An attacker can generate a new address with any subnet prefix. CGA does not prove that the node or address exists
- Attacks against link layer may be just as bad

CGA advantages

- Authentication of an IP address without a PKI or other security infrastructure
- With Secure DNS, gives strong host authentication
- Without Secure DNS, prevents many DoS attacks
- Particularly suitable for authenticating IP-layer signaling

SEND

- Secure neighbor discovery (SEND) [RFC 3971]
- CGA-based signatures on neighbor advertisements
  - Prevents NA spoofing
  - Prevents address squatting in DAD
  - Zero-configuration security!
- Certificate-based authorization of routers
  - Certificate authorizes router for a an address prefix
  - Extension to X.509 to certify IPv6 address allocation [RFC 3779]
  - Requires hosts to know the root key; currently no global CA hierarchy
- Freshness:
  - Timestamp in unsolicited advertisement and redirect
  - Nonce in NS and RS, copied to NA and RA

PAIRING

Device pairing

- Device pairing: establishing a shared key between two devices
  - Out-of-band authentication: user decides which devices to pair and has physical access to them
  - No device names or credentials required, although they may help
  - One device may be assigned a name in the other's name space
  - One device may become master/owner of the other
- Device-to-cloud registration: paring with a cloud service

Basic pairing methods

1. Out-of-band (OOB) input of a strong secret to both devices
   - Can be used directly or to authenticate Diffie-Hellman
2. Transmitting secret key out-of-band
   - OOB channel must protect secrets
3. Diffie-Hellman + OOB comparison of key hashes
   - OOB channel must protect integrity

- Variants with subtle differences:
  a. Out-of-band transmission of hashes in both directions and comparison at both devices
  b. Human comparison and "ok" input to both devices
  c. Out-of-band transmission in one direction, comparison by the receiver, and "ok" input to the sender
- Secret or hash must be at least 128 bits. Why?

Advanced pairing protocols

- Goal: better usability
  - OOB input of a weak secret to both devices: PIN code or weak password
  - OOB comparison or input of shorter messages: ~20 bits (6 digits) instead of 128 bits (~38 digits)
- Example: Bluetooth secure simple pairing

Bluetooth SSP wth hash comparison

- ECDH:

$$A \rightarrow B\colon\ PK_a$$
$$B \rightarrow A\colon\ PK_b$$

- A and B generate nonces. B commits to its nonce for "fairness":

$$B \rightarrow A\colon\ C_A = f_1(PK_b, PK_a, N_b, 0)$$
$$A \rightarrow B\colon\ N_a$$
$$B \rightarrow A\colon\ N_b$$

- A checks the commitment $C_a =^{?} f_1(PK_b, PK_a, N_b, 0)$
- Both A and B display and user compares:

$$v_A = g(PK_a, PK_b, N_a, N_b)$$
$$v_B = g(PK_a, PK_b, N_a, N_b)$$

- User confirms "ok" on both devices
- Authentication:

$$A \rightarrow B\colon\ E_a = f_3(DHKey, N_a, N_b, IOcapA, A, B)$$
$$B \rightarrow A\colon\ E_b = f_3(DHKey, N_b, N_a, IOcapB, B, A,)$$

- Both compute link key:

$$LK = f_2(DHKey, N_{master}, N_{slave}, \text{"btlk"}, ADDR_{master}, ADDR_{slave})$$

Why is a 6-digit code sufficient? How did the commitment help?

Exercises

- Can you design a secure key exchange protocol for connecting Wi-Fi speakers to a home computer based on:
  - Trusted hub device e.g. the network gateway
  - User-verified key exchange
  - Location-limited audio channel
  - Leap of faith
  - Self-certifying identifiers
- What are the weaknesses in each solution?
- Learn about the different Bluetooth pairing protocols
- Learn about the authentication of location updates in Mobile IPv6